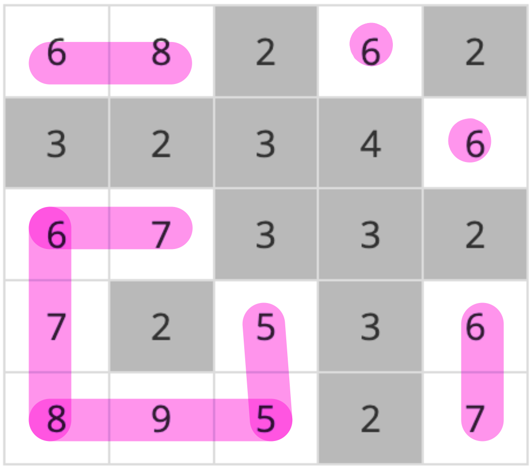
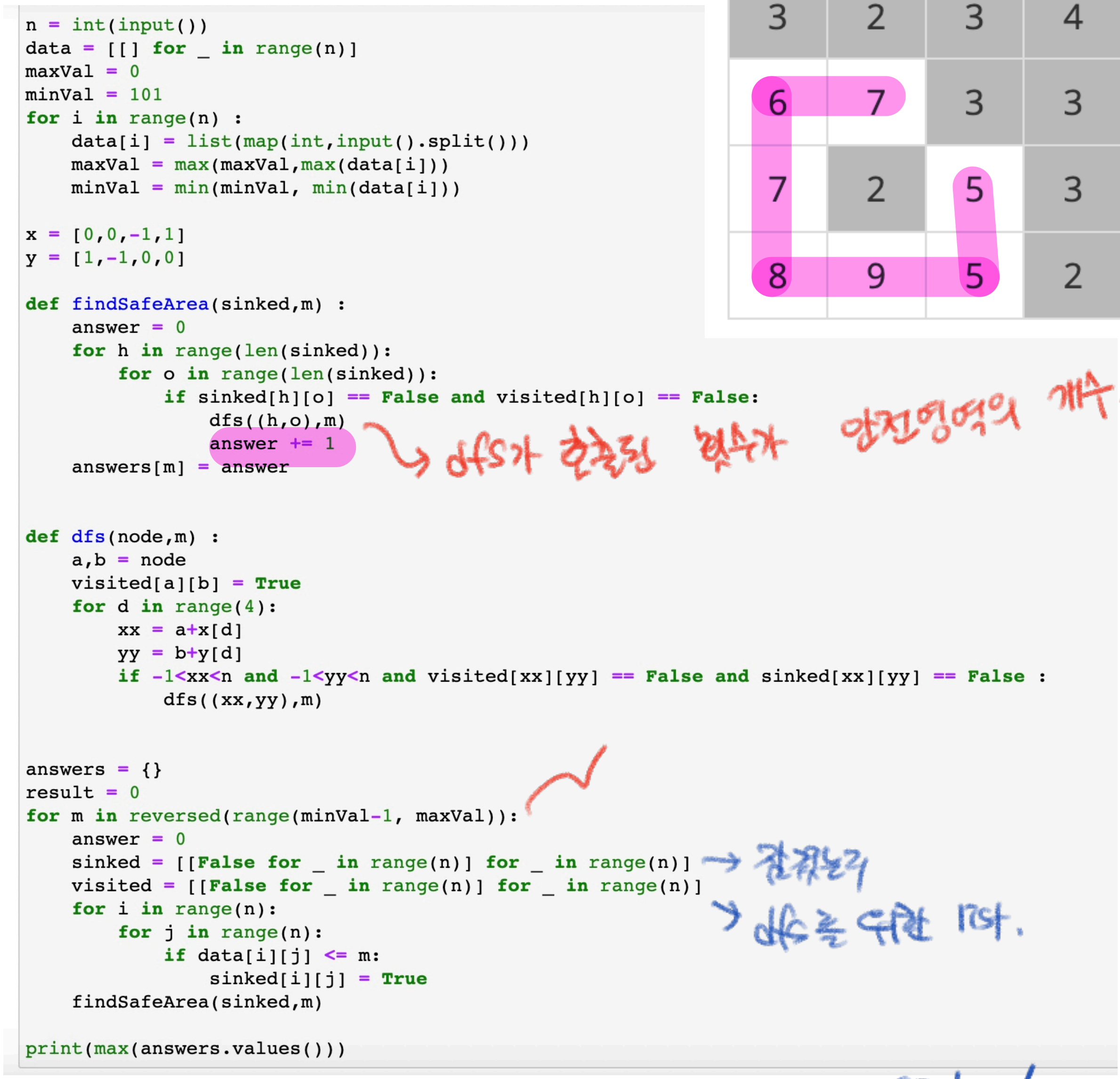


6	8	2	6	2
3	2	3	4	6
6	7	3	3	2
7	2	5	3	6
8	9	5	2	7



```python
n = int(input())
data = [[] for _ in range(n)]
maxVal = 0
minVal = 101
for i in range(n) :
    data[i] = list(map(int,input().split()))
    maxVal = max(maxVal,max(data[i]))
    minVal = min(minVal, min(data[i]))

x = [0,0,-1,1]
y = [1,-1,0,0]

def findSafeArea(sinked,m) :
    answer = 0
    for h in range(len(sinked)):
        for o in range(len(sinked)):
            if sinked[h][o] == False and visited[h][o] == False:
                dfs((h,o),m)
                answer += 1
    answers[m] = answer


def dfs(node,m) :
    a,b = node
    visited[a][b] = True
    for d in range(4):
        xx = a+x[d]
        yy = b+y[d]
        if -1<xx<n and -1<yy<n and visited[xx][yy] == False and sinked[xx][yy] == False :
            dfs((xx,yy),m)


answers = {}
result = 0
for m in reversed(range(minVal-1, maxVal)):
    answer = 0
    sinked = [[False for _ in range(n)] for _ in range(n)]
    visited = [[False for _ in range(n)] for _ in range(n)]
    for i in range(n):
        for j in range(n):
            if data[i][j] <= m:
                sinked[i][j] = True
    findSafeArea(sinked,m)

print(max(answers.values()))
```

↳ dfs가 호출된 횟수가 안전영역의 개수.

→ 잠겼는지

→ dfs를 위한 list.

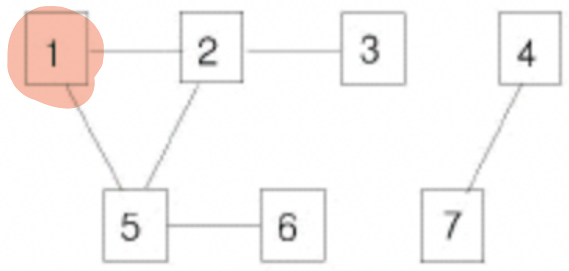
1. min ~ max ⓜ 잠겼는지 . sinked

2. sinked 가지고 False [안전지대] 안전영역 chk

↳ find Safe Area.

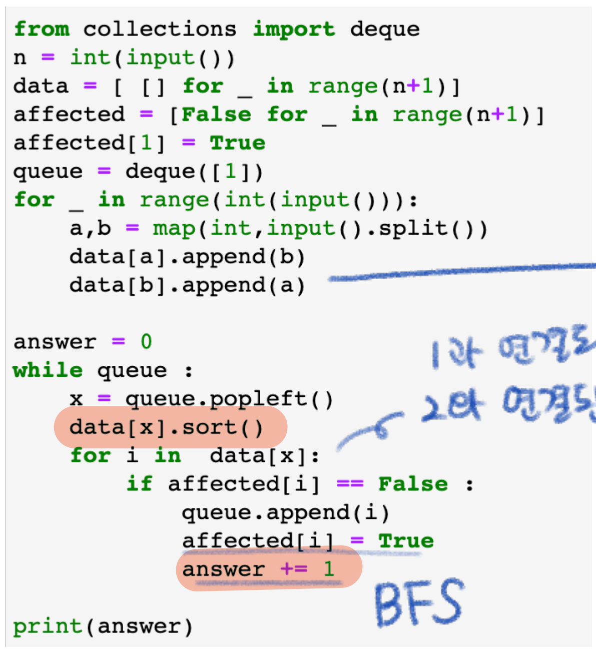
2606 바이러스

- 1번이 바이러스에 걸렸을 때, 1번 컴퓨터를 통해 감염되는 컴퓨터 수.



< 그림 1 >

bfs도 dfs도 [illegible] 전체방문만 하면 됨.



```
from collections import deque
n = int(input())
data = [ [] for _ in range(n+1)]
affected = [False for _ in range(n+1)]
affected[1] = True
queue = deque([1])
for _ in range(int(input())):
    a,b = map(int,input().split())
    data[a].append(b)
    data[b].append(a)

answer = 0
while queue :
    x = queue.popleft()
    data[x].sort()
    for i in  data[x]:
        if affected[i] == False :
            queue.append(i)
            affected[i] = True
            answer += 1

print(answer)
```

1과 연결된..

2와 연결된..

BFS

data

[ ]

[2 . 5]

[2]

[7]

[1 , 2]

[5]

[4]

DFS